Some Software Design Considerations For Targeted Adaptive Design

Carlo Graziani

July 28, 2021

1 Introduction

This memo is intended to aid in the transition from the mathematical abstraction of the "Targeted Adaptive Design" (*TAD*) memo to an actual practical software design that is potentially scalable and extensible. The design criteria that I have in mind are that certain mathematical elements of the problem can be clearly and naturally isolated into their own software modules, and that a certain software hierarchy should naturally emerge from the mathematical problem. I hope that what I mean by these admittedly obscure remarks will become more clear below.

2 Description of the Algorithm

2.1 Design Space and Data

We are given a "design space" $X \subset \mathbb{R}^D$, which will usually be a D-dimensional hypercube of experimental "knobs" that can be set on some manufacturing apparatus. There is an unknown response function $f(\cdot)$ on the design space $f : X \to Y \subset \mathbb{R}^E$. The elements of Y represent features of a manufactured product that we would like to produce — for example, sizes and smoothnesses of certain planar surfaces, angles between planar surfaces, material compositions, and so on. So obviously the function $f(\cdot)$ represents the mapping from design apparatus settings to final product. We would like to find a setting $\boldsymbol{x}_T \in X$ that results in a target product $\boldsymbol{y}_T$ within some tolerance region — that is, such that $f(\boldsymbol{x}_T) \approx \boldsymbol{y}_T$ within the tolerance. We will see below how the tolerance is to be specified.

The problem is that it is a complicated mapping, and initially we don't really know what it is. So the plan is that we will impose a vector-valued GP model on $f(\cdot)$, and perform sequential sets of experiments, where at the μth stage we perform N_μ experiments at experimental settings $\boldsymbol{X}^{(\mu)} \equiv \left\{\boldsymbol{x}_k^{(\mu)} \in X; k = 1, \ldots, N_\mu\right\}$, obtaining N_μ noisy measurements $Y^{(\mu)} \equiv \left\{\boldsymbol{y}_k^{(\mu)} = f(\boldsymbol{x}_k^{(\mu)}) + \epsilon_k^{(\mu)}; k = 1, \ldots, N_\mu\right\}$ contaminated by additive zero-mean gaussian noise $\epsilon_k^{(\mu)} \sim \mathcal{N}\left(0, \Sigma_k^{(\mu)}\right)$, with noise covariance $\Sigma_k^{(\mu)}$ diagonal in the most common case, since it is usually the case that the components of the measurement errors are i.i.d.[1] Let's also aggregate the data already "in the can" at settings $\Xi^{(\mu)} \equiv \cup_{\nu=1}^{\mu} \boldsymbol{X}^{(\nu)}$, with observations $\Upsilon^{(\mu)} \equiv \cup_{\nu=1}^{\mu} \boldsymbol{Y}^{(\nu)}$.

Note that we have abandoned here the generality of the "measurement operator" formulation in *TAD*, assuming that we will only use the "point sample" operator G_x. The more general formulation is interesting, but not useful in the current context, and is partly responsible for the notational snarl in *TAD*.

[1] This is not a necessary restriction, however.

2.2 Gaussian Process Model

The GP model specifies a matrix-valued kernel and a vector-valued mean function, and is characterized by certain hyperparameters α. We will hence write the kernel $K(\boldsymbol{x}, \boldsymbol{x}'; \alpha)$ and the mean function $m(\boldsymbol{x}; \alpha)$.

Note that as a consequence of the GP parametrization, there are *two* optimization problems to be solved in this problem, even though *TAD* only discussed one: there is the optimization of future experimental settings $\boldsymbol{X}^{(\mu+1)}$ given data available at stage μ — this is described in *TAD*. There is also the optimization of the GP parameters $\alpha^{(\mu)}$ at the stage μ given the currently available data, which must precede the optimization over $\boldsymbol{X}^{(\mu+1)}$.

The kernel $K(\cdot)$ is a matrix, with matrix elements $[K(\boldsymbol{x}, \boldsymbol{x}'; \alpha)]_{ll'}$, $l, l' = 1, \ldots, E$. It must furnish a positive-definite kernel over the space of E-dimensional vector fields over X. A few possible types of such kernels are discussed in [1]. They are sufficiently different that there will be a software design challenge associated with making a few desirable models available — different types offer different optimizations, for example, so some care must be taken in associating different methods with them.

The first, simplest type to implement will be the tensor product,

$$[K(\boldsymbol{x}, \boldsymbol{x}'; \alpha)]_{ll'} \equiv k(\boldsymbol{x}, \boldsymbol{x}'; \alpha) \times \kappa_{ll'}, \tag{2.1}$$

where $k(\cdot)$ is an arbitrary scalar GP kernel, and κ is a positive-definite $E \times E$ matrix. This form is particularly convenient because the "inversions" (really, linear solves, one never inverts of course) can be performed separately in the two product spaces, resulting in great computational savings. It is particularly convenient because we can choose as a parametrization of the matrix factor κ the Cholesky decomposition of κ, that is, an arbitrary $E \times E$ lower-triangular matrix ψ (so, $E(E-1)/2$ parameters), and set $\kappa = \psi\psi^T$, which is obviously positive-definite. Thus, not only does the associated linear problem factor, but the linear problem associated with the κ matrix is already solved, since we already have the Cholesky factors!

Note that the Cholesky decomposition is unique so long as the diagonal elements of the factors are positive-definite, so this is a restriction on the parametrization.

Another possible interesting matrix-valued kernel arises from the optimization problem, where we attempt to find stationary points of an experimentally-discovered scalar field $\phi(\cdot)$ by zeroing its gradient, that is, by finding points where $\nabla\phi = 0$. In this case we have $E = D$, and after placing a scalar GP model on $\phi(\cdot) \sim \mathrm{GP}(k(\cdot,\cdot), m(\cdot))$ we model $\nabla\phi$ using the matrix kernel

$$[K(\boldsymbol{x}, \boldsymbol{x}'; \alpha)]_{ll'} \equiv \frac{\partial}{\partial x_l} \frac{\partial}{\partial x'_{l'}} k(\boldsymbol{x}, \boldsymbol{x}'; \alpha), \tag{2.2}$$

which can also be shown positive-definite by integration by parts. In this case we will not have nice factorizations of linear problems into tensor factors, however.

As to mean vectors, the common practice of setting them summarily to zero is generally a bad idea, because it produces models that sag back to zero as soon as they escape the influence of the data (i.e. they are biased). This is generally avoided by the practice of mean-subtracting the data, which in my opinion is not great, since it leads to an annoying accounting issue in which the data means need to be kept track of and put back in at various stages. The most convenient basic approach that I have found is to use a constant mean function,

$$m(\boldsymbol{x}, \alpha) \equiv m_0, \tag{2.3}$$

where m_0 is a constant in space and across the E components of $m(\cdot)$ (and is one of the parameters in the list α). When one does this, one finds that GP optimization over the parameter m_0 can in general be performed analytically, resulting in a formula of the type

$$m_0 = \frac{\mathbf{1}^T \boldsymbol{C}^{-1} \boldsymbol{f}}{\mathbf{1}^T \boldsymbol{C}^{-1} \mathbf{1}}, \tag{2.4}$$

where $\mathbf{1}$ is a vector whose components are all 1, $\boldsymbol{f}$ is the data — a list of E-dimensional vectors, one at each location in X, and $\boldsymbol{C}$ is the covariance describing the data. That is, m_0 turns out to be a weighted mean of the data, and this method naturally "mean-subtracts" the data inside the model. I would like the software to at least have the option to do this.

Clearly, however, in the optimization case, a constant mean function for the scalar function $\phi(\cdot)$ leads to a zero mean for the gradient $\nabla\phi(\cdot)$, so in some cases we do want to have to treat zero-mean GPs.

2.3 Transition Between Full and Stripped-Down Formulations

For the next part of the discussion, a little more notation is useful. Let's introduce the "present+future" covariance matrix from Equation (4.1) of *TAD,*

$$C \equiv \begin{bmatrix} C_{ff} & C_{f1} & C_{f1} \\ C_{1f} & C_{11} & C_{12} \\ C_{2f} & C_{21} & C_{22} \end{bmatrix}, \tag{2.5}$$

the "present-only" covariance

$$C' \equiv \begin{bmatrix} C_{ff} & C_{f1} \\ C_{1f} & C_{11} \end{bmatrix}, \tag{2.6}$$

and the "present data-only" covariance C_{11}.

Also, the predictive log-likelihood at a point $\boldsymbol{x}$ as a function of target vector f_T, covariance matrix $Q(\boldsymbol{x})$ and mean vector $p(\boldsymbol{x})$,

$$L(f_T, Q, p) \equiv -\frac{1}{2}\log\det(Q) - \frac{1}{2}(f_T - p)^T (Q)^{-1}(f_T - p), \tag{2.7}$$

and the trace term from Equation (4.34) of *TAD*,

$$T(C) \equiv -\frac{1}{2}\text{Trace}\left\{\left(C_{f2} - C_{f1}C_{11}^{-1}C_{12}\right)\left(Q^{(2|1)}\right)^{-1}\left(C_{2f} - C_{21}C_{11}^{-1}C_{1f}\right)\left(Q^{(f|1+2)}\right)^{-1}\right\}, \tag{2.8}$$

where $Q^{(2|1)}$ and $Q^{(f|1+2)}$ are related to C by Equations (4.16) and either (4.22) or (4.30)+(4.19) of *TAD*.

Equations (2.5-2.8) are expressed in terms of the "stripped-down" notation, which knows only the Gaussian structure of the problem, but knows nothing of the experimental structure. The bridge between the two is constructed by means of mappings

$$C \equiv M\left(\Xi^{(\mu+1)}, \boldsymbol{x}; \alpha^{(\mu)}\right), \tag{2.9}$$

$$C' \equiv M\left(\Xi^{(\mu)}, \boldsymbol{x}; \alpha^{(\mu)}\right), \tag{2.10}$$

$$C_{11} \equiv N\left(\Xi^{(\mu)}; \alpha^{(\mu)}\right), \tag{2.11}$$

which are obviously redundant in consequence of the nestedness of C' in C. These mappings are essentially as described in Equations (4.36-4.41) of *TAD*, with the generality of the "measurement operators" $G(\theta)$ replaced by the assumption that we are using the sampling operator G_x (so that the kernel $K(\cdot,\cdot)$ is evaluated at appropriate pairs of points). Similar remarks apply to the mapping of mean vectors to their stripped-down versions.

Equation (4.29), (4.44), and (4.45) of *TAD*, give other required functional relationships that exploit C and/or C':

$$p^{(f|1)} \equiv P\left(\Xi^{(\mu)}, \Upsilon^{(\mu)}, \boldsymbol{x}; \alpha^{(\mu)}\right); \tag{2.12}$$

$$Q^{(f|1)} \equiv R\left(\Xi^{(\mu)}, \boldsymbol{x}; \alpha^{(\mu)}\right); \tag{2.13}$$

$$Q^{(f|1+2)} \equiv S\left(\Xi^{(\mu+1)}, \boldsymbol{x}; \alpha^{(\mu)}\right). \tag{2.14}$$

Constructing the mappings $M(\cdot)$, $P(\cdot)$, $R(\cdot)$, and $S(\cdot)$ is clearly an important part of the software task. It also seems to me a natural place for a hierarchy interface between an "upper" layer that is conscious of experimental detail and a "lower" layer that knows only Gaussian model elements.

2.4 Objective Functions

At the beginning of stage μ, we have just gathered new observations $Y^{(\mu)}$ at the settings $X^{(\mu)}$ proposed at the previous $(\mu-1)$ stage. It is time to optimize GP hyperparameters $\alpha^{(\mu)}$ affecting kernel and mean function. We do this by maximizing the log-likelihood

$$\hat{\mathcal{L}}\left(\alpha^{(\mu)}\right) \equiv -\frac{1}{2}\log\det C_{11} - \frac{1}{2}\left(\Xi^{(\mu)} - m_0\mathbf{1}\right)^T C_{11}^{-1}\left(\Xi^{(\mu)} - m_0\mathbf{1}\right) \tag{2.15}$$

with respect to $\alpha^{(\mu)}$. Note that this is a different type of likelihood from the one optimized in *TAD* — it is the classic GP likelihood of all the (current at μ) data, rather than a predictive likelihood of the target f_T at a point $\boldsymbol{x}$. As noted above, it is a simple algebraic exercise to show that this expression is maximized with respect to m_0 by the choice

$$m_0 = \frac{\mathbf{1}^T C_{11}^{-1} \Xi^{(\mu)}}{\mathbf{1}^T C_{11}^{-1} \mathbf{1}}. \tag{2.16}$$

At this point, we have an optimal GP model given data "in the can". We propose $N_{\mu+1}$ new points at settings $\boldsymbol{X}^{(\mu+1)}$, and choose them by optimizing the expected log likelihood

$$\mathcal{L}\left(\boldsymbol{X}^{(\mu+1)}, \boldsymbol{x}\right) \equiv L\left(f_T, Q^{(f|1+2)}, p^{(f|1)}\right) + T\left(C\right) \tag{2.17}$$

with respect to $\boldsymbol{X}^{(\mu+1)}$ and $\boldsymbol{x}$, where $L(\cdot)$ and $T(\cdot)$ are the functions defined in Equations (2.7) and (2.8).

2.5 Stopping

There have to be two stopping conditions: success (found target within tolerance) and failure (no target within tolerance). We will make the stopping decision after the new data acquisition and optimization over GP hyperparameters $\alpha^{(\mu)}$, but before optimising new proposed settings $\boldsymbol{X}^{(\mu+1)}$.

We introduce an E-dimensional tolerance vector τ with components τ_i, $i = 1, \ldots, E$ representing the tolerance within which we must hit each component of the target.

Define intervals $\eta_i \equiv [f_{Ti} - \tau_i, f_{Ti} + \tau_i]$, $i = 1, \ldots, E$ (where f_{Ti} is the i-th component of the target vector f_T). This is the "targeted tolerance region.

Now define the "1-σ" intervals $\xi_i(\boldsymbol{x}) \equiv \left[p_i^{(f|1)}(\boldsymbol{x}) - \left(Q_i^{(f|1)}(\boldsymbol{x})\right)^{1/2}, p_i^{(f|1)}(\boldsymbol{x}) + \left(Q_i^{(f|1)}(\boldsymbol{x})\right)^{1/2}\right]$, $i = 1, \ldots, E$. This is our measure of uncertainty of our vector function along each dimension.

Initially, we choose prior GP parameters α in such a way that the $\left(Q_i^{(f|1)}\right)^{1/2}$ are quite large, expressing our ignorance of the function $f(\cdot)$. As a consequence, we should start in a state where $\forall_{\boldsymbol{x}\in X} \forall_{i\in[1\cdots E]} \left(\eta_i \subset \xi_i(\boldsymbol{x})\right)$.

The "Success" criterion that we test for is this: at the $\boldsymbol{x}$ value located by the most recent optimization, $\forall_{i\in[1\cdots E]} \left(\xi_i(\boldsymbol{x}) \subset \eta_i\right)$. This means "The optimum 1-σ uncertainty region is entirely contained within the targeted tolerance region."

The "Failure" criterion that we test for is this: at the $\boldsymbol{x}$ value located by the most recent optimization, $\exists_{i\in[1\ldots E]} \left(\xi_i(\boldsymbol{x}) \bigcap \eta_i = \emptyset\right)$. This means "even the at the optimum parameter settings, the 1-σ uncertainty region is disjoint from the targeted tolerance region."

If neither "Success" or "Failure" criteria are met, we continue to the next iteration.

In order for this to work correctly, some care must be taken to ensure robust convergence to a global maximum of $\boldsymbol{X}^{(\mu)}, \boldsymbol{x}$, so as not to be fooled by local "Failure," and so as not to miss a "Success."

References

[1] Mauricio A Alvarez, Lorenzo Rosasco, and Neil D Lawrence. Kernels for vector-valued functions: A review. *arXiv preprint arXiv:1106.6251*, 2011.